AUSTRALIAN OS9 NEWSLETTER

sides 'No. of cylinders' (in decimal) :Interleave value: (in decimal) @FREE Syntax: Free [devname] Usage : Displays number of free sectors on a device @GFX Syntax: RUN GFX(‹funct›‹args›) Usage : Graphics interface package for BASIC09 to do compatible VDG graphics commands @GFX2 Syntax: RUN GFX2([path]‹funct›‹args›) Usage : Graphics interface package for BASIC09 to handle ax: none Usage : graphics/ windows on-line help to topics @IDENT mation from OS = use single lin ecution directory device @INKEY utine to input a ort to the pro k to a memory ntents of text files @LOAD Syntax: Load ‹pathname› [...] Usage : Loads modules into memory @MAKDIR Syntax: Makdir ‹pathname› Usage : Creates a new directory file @MDIR Syntax: Mdir [e] Usage : Displays the present memory module directory Opts : e = print extended module directory @MERGE Syntax: Merge ‹path› ndard output @MFREE Synt RAM memory @MODPATCH a module in memory from warnings -c = compare modul name = link to module C o e V = verify module M = ma Montype [opt] Usage : Set mo monitor m = monochrome m sage : Creates and links an OS ROCS Syntax: Procs [e] Usage em Opts : e = display all proc Prints the current data directory path @PXD Syntax: Pxd Usage : Prints the current execution directory path @RENAME Syntax: Rename ‹filename› ‹new filename› Usage : Gives the file or directory a new name @RUNB Syntax: Runb ‹i-code module› Usage : BASIC09 run time package @SETIME Syntax: Setime [yy/mm ETPR Syntax: ss to num @ reter @TMODE Syntax: Tmode [pathname] [params] Usage : Displays or changes the operating parameters of the terminal @TUNEPORT Tuneport ‹/t1 or /p› [value] Adjust the baud value for the serial port @UNLINK Syntax: Unlink ‹modname› Usage : Unlinks module(s) from memory @WCREATE Syntax:

EDITOR	Gordon Bentzen	(07) 344-3881
SUB-EDITOR	Bob Devries	(07) 372-7816
TREASURER	Don Berrie	(07) 375-1284
LIBRARIAN	Jean-Pierre Jacquet	(07) 372-4675
	Fax Messages	(07) 372-8325
SUPPORT	Brisbane OS9 Users Group	

Addresses for Correspondence

Editorial Material:

Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

Subscriptions & Library Requests:

Jean-Pierre Jacquet
27 Hampton Street
DURACK Qld 4077

Volume 5 October 1991 Number 9

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 5 Number 9

EDITOR : Gordon Bentzen
SUBEDITOR : Bob Devries
TREASURER : Don Berrie
LIBRARIAN : Jean-Pierre Jacquet
SUPPORT : Brisbane OS9 Level 2 Users Group.

Well folks, by the time that you get this, Gordon and his wife Val will be winging their way to Europe for a well earned holiday. This means that the quality of the editorials will be somewhat lower than is usual for our Newsletter.

Nevertheless, we do have some goodies for you this month. Our articles once again include some discussions from the Internet CoCo OS9 Mailing List. Apologies to those readers who already receive those articles by electronic means. If you have access to E-Mail, but don't subscribe to the List, we have included some information to help you to subscribe to this list.

For those of you who have an interest, even if it's only a hypothetical one, in OSK, we have been able to get hold of a relatively large amount of OSK public domain software from EFFO. EFFO is an acronym for the European Forum For OSK, and is mainly comprised of European Users of OSK in Atari ST's. Most of their software however, is not machine specific, so should be usable by most, if not all OSK users.

The software is contained in large (approx 250K-350K) archives of public domain disks (about 9), and larger (approx 450K) archives of forum disks (about 17). At present they are only available in the form of *.lzh archives, as they all dearchive to files of a size greater than 720K!! This obviously rather limits their usefulness to those who can unarchive them themselves, or to those who have access to 1.44 Mb disk drives. Unfortunately, at present, we do not have these drives (for OS9/K), and the only way we can get at them, is to unarchive them directly to a hard drive.

Anyway, we are working on a way to split the archives into multiple smaller archives, and perhaps will be able to distribute them in that form to those who are interested. This method will require the permission of the EFFO people, and we will be attempting to do that.

We have included a very brief contents listing of the EFFO public domain disks for your information.
We have received some further additions to out public domain library. One of those archives allows patching of the CC3Disk driver to allow for disk caching, using up to 100 (?) sectors in the cache. This may be of interst to those of you who do not have no-halt controllers.

Another archive supplies an ipatch file to patch the OS9 kernal to allow some of the "features" of the OSK file naming conventions to be implemented. Filenames starting with non-alphabetic characters, for example, are allowable, as are filenames which do not contain any alphabetic characters.

These archives are available from our librarian, Jean-Pierre. The usual charges will apply.

Enough of the editorial,

Cheers - Don Berrie (for Gordon)

oooooooooo0000000000ooooooooo

CoCo-Link

CoCo-Link is an excellent magazine to help you with the RSDOS side of the Colour Computer. It is a bi-monthly magazine published by Mr. Robbie Dalzell. Send your subscriptions to:

CoCo-Link
31 Hedlands Crescent
Pt. Noarlunga Sth.
South Australia
Phone: (08) 3861647

A Basic09 Tutorial
by Bob Devries

Basic09 has always been a much mis-understood language. I suppose that is because most Basic09 users come from Disk Basic, and the change is quite marked, especially the lack of line numbers. It does, however, have a few quite powerful looping commands which can be used to make up for the lack(?). These looping commands are: FOR / NEXT / STEP...aha, I see you recognize that one...IF / THEN / ELSE / ENDIF ...you almost know that one too... LOOP / ENDLOOP ...a new one.. REPEAT / UNTIL ..and.. WHILE / DO / ENDWHILE ..and..EXITIF / ENDEXIT.

The FOR/NEXT loop is much the same as the RSBASIC one. The loop variable may be either an INTEGER or REAL, and the STEP value may be positive or negative. So far that's easy.

Now comes the IF/THEN loop. The new word here is the ENDIF statement. Because Basic09 does not (usually) use line numbers, there must be a way to tell the interpreter where the IF/THEN loop finishes. This is where it is different from RSBASIC. With Basic09, the IF/THEN loop can be, and usually is, split over multiple lines. Here's a sample of programme to show you.

```
IF a$="2" THEN
   RUN gfx2("bell")
   print "You pressed '2'."
ELSE
   RUN gfx2("bell")
   print "You pressed something else."
ENDIF
```

So you can see that the ENDIF is required, so that Basic09 can see where to start again after the THEN part has been done. In RSBASIC, this would all have to be on one line! So you can see how much more powerful it is!

The next loop command is called just that: LOOP and ENDLOOP. This forms an unconditional branch back from the ENDLOOP to the LOOP. The only way out of this loop is to use the EXITIF/ENDEXIT command. This allows you to exit any loop prematurely, and is particularly useful here. Here's how it loops..er looks:

```
LOOP
   INPUT ")",a$
   EXITIF a$="exit" THEN
   ENDEXIT
   PRINT a$
ENDLOOP
```

The reason the LOOP/ENDLOOP is used here is because the other looping commands all go through the loop at least once before they terminate, or require that the test condition is known beforehand. As this is not the case here, I have used the LOOP/ENDLOOP command.

If you already have the starting value of the 'condition' variable, then the WHILE/DO/ENDWHILE command is useful. Here is an example:

```
DIM a,b:STRING

WHILE a="" DO
   RUN inkey(a)
   print a;
   b=b+a
ENDWHILE
```

Here the loop may not be used at all, depending on the value of the string variable 'a'. If 'a' id a NULL string (no characters), then the loop contents are executed. Otherwise, they are not, and the programme starts again after the ENDWHILE command. If you must do the contents of a loop at least once, use this method:

```
REPEAT
  a=PEEK($FF40)
UNTIL LAND(a,8)=0
```

This way, the value of a is the result of a PEEK command, and is not known until at least once around the loop. If the value makes the UNTIL argument equal zero, then the loop is only executed once, otherwise it goes round again.

Loops may be nested, of course, and different loops may be used within each other to make the code more like spaghetti. At least that is what it might look like to a beginner. At least Basic09, when asked to list the programme, inserts indents for the various levels of nesting so that you can see at a glance what is going on. And of course, if you forget an ENDIF, or other end-of-loop statement, it tells you in no uncertain terms that you have erred! Error 69 - Unmatched control structure means you have left out an ENDIF or ENDLOOP or something.

Of course, GOTO (yuk) can be used if you have

used a line number to go to. I guess it is also a form of loop, and can be used within the loop structure to by-pass bits of code. Line numbers are also used if you want to used GOSUBs. Basic09 does not use text labels, but uses line numbers instead. Other forms of loop or control structures are ON...GOTO, ON...GOSUB, and ON ERROR. These all need line numbers to refer to. The ON ERROR is particularly useful in trapping user errors, but don't use them for trapping programme errors (bugs), or you may never see them! Always REM them out until you have thoroughly debugged your code, and only unREM them to trap unwary users of your programme.

Well, I hope that gave you something to go on with. Until next time...

Regards, Bob

oooooooooo0000000000ooooooooooo

A C Tutorial
Chapter 4 - Assignment & Logical compares

INTEGER ASSIGNMENT STATEMENTS

Load the file INTASIGN.C and display it for an example of assignment statements. Three variables are defined for use in the program and the rest of the program is merely a series of illustrations of various assignments. The first two lines of the assignment statements assign numerical values to "a" and "b", and the next four lines illustrate the five basic arithmetic functions and how to use them. The fifth is the modulo operator and gives the remainder if the two variables were divided. It can only be applied to "int" or "char" type variables, and of course "int" extensions such as "long", "short", etc. Following these, there are two lines illustrating how to combine some of the variables in some complex math expressions. All of the above examples should require no comment except to say that none of the equations are meant to be particularly useful except as illustrations. The next two expressions are perfectly acceptable as given, but we will see later in this chapter that there is another way to write these for more compact code.

This leaves us with the last two lines which may appear to you as being very strange. The C compiler scans the assignment statement from right to left, (which may seem a bit odd since we do not read that way), resulting in a very useful construct, namely the one given here. The compiler finds the value 20, assigns it to "c", then continues to the left finding that the latest result of a calculation should be assigned to "b". Thinking that the latest calculation resulted in a 20, it assigns it to "b" also, and continues the leftward scan assigning the value 20 to "a" also. This is a very useful construct when you are initializing a group of variables. The last statement illustrates that it is possible to actually do some calculations to arrive at the value which will be assigned to all three variables. The program has no output so compiling and executing this program will be very uninteresting. Since you have already learned how to display some integer results using the "printf" function, it would be to your advantage to add some output statements to this program to see if the various statements do what you think they should do. This would be a good time for a preliminary definition of a rule to be followed in C. The data definitions are always given before any executable statements in any program block. This is why the variables are defined first in this program and in any C program. If you try to define a new variable after executing some statements, the compiler will issue an error.

ADDITIONAL DATA TYPES

Loading and editing MORTYPES.C will illustrate how some additional data types can be used. Once again we have defined a few integer type variables which you should be fairly familiar with by now, but we have added two new types, the "char", and the "float". The "char" type of data is nearly the same as the integer except that it can only be assigned values between zero and 255, since it is stored in only one byte of memory. The "char" type of data is usually used for ASCII data, more commonly known as text. The text you are reading was originally written on a computer with a word processor that stored the words in the computer one character per byte. In contrast, the integer data type is stored in two bytes of computer memory on most microcomputers.

DATA TYPE MIXING

It would be profitable at this time to discuss the way C handles the two types "char" and "int". Most functions in C that are

designed to operate with integer type variables will work equally well with character type variables because they are a form of an integer variable. Those functions, when called on to use a "char" type variable, will actually promote the "char" data into integer data before using it. For this reason, it is possible to mix "char" and "int" type variables in nearly any way you desire. The compiler will not get confused, but you might. It is good not to rely on this too much, but to carefully use only the proper types of data where they should be used. The second new data type is the "float" type of data, commonly called floating point data. This is a data type which usually has a very large range, a large number of significant digits, and a large number of computer words are required to store it. The "float" data type has a decimal point associated with it and, on most computers, has an allowable range of from 10E-38 to 10E+38. Not all compilers have the same available range, so check your reference manual for the limits on your compiler.

HOW TO USE THE NEW DATA TYPES

The first three lines of the program assign values to all nine of the defined variables so we can manipulate some of the data between the different types. Since, as mentioned above, a "char" data type is in reality an "integer" data type, no special considerations need be taken to promote a "char" to an "int", and a "char" type data field can be assigned to an "int" variable. When going the other way, there is no standard, so you may simply get garbage if the value of the integer variable is outside the range of the "char" type variable. It will translate correctly if the value is within the range of zero to 255. In the second line therefore, when attempting to set x (a char) to -27, you may or may not get a well defined answer, it depends on your particular implementation of C. The third line illustrates the simplicity of translating an integer into a "float", simply assign it the new value and the system will do the proper conversion. When going the other way however, there is an added complication. Since there may be a fractional part of the floating point number, the system must decide what to do with it. By definitions , it will truncate it. This program produces no output, and we haven't covered a way to print out "char" and "float" type variables, so you can't really get in to this program and play with the results, but the next program will cover this for you.

LOTS OF VARIABLE TYPES

Load the file LOTTYPES.C and display it on your screen. This file contains every standard simple data type available in the programming language C. There are other types, but they are the compound types that we will cover in due time. Observe the file. First we define a simple "int", followed by a "long int" and a "short int". Consult your reference manual for an exact definition of these for your compiler, because they are not consistent from implementation to implementation. The "unsigned" is next and is defined as the same size as the "int" but with no sign. The "unsigned" then will cover a range of 0 to 65535 on most microcomputers. It should be pointed out that when the "long", "short", or "unsigned" is desired, the "int" is optional and is left out by most experienced programmers. We have already covered the "char" and the "float", which leaves only the "double". The "double" usually covers a greater range than the "float" and has more significant digits for more precise calculations. It also requires more memory to store a value than the simple "float". Consult your reference manual for the range and accuracy of the "double".

Another diversion is in order at this point. Most compilers have no provisions for floating point math, but only double floating point math. They will promote a "float" to a "double" before doing calculations and therefore only one math library will be needed. Of course, this is totally transparent to you, so you don't need to worry about it. You may think that it would be best to simply define every floating point variable as double, since they are promoted before use in any calculations, but that may not be a good idea. A "float" variable requires 4 bytes of storage and a "double" requires 8 bytes of storage, so if you have a large volume of floating point data to store, the "double" will obviously require much more memory. Your compiler may require a different number of bytes than 4 or 8. Consult your reference manual for the correct number of bytes used by your compiler.

After defining the data types, a numerical value is assigned to each of the defined variables in order to demonstrate the means of outputting each to the monitor.

THE CONVERSION CHARACTERS

Following is a list of the conversion characters and the way they are used in the

"printf" statement.

d decimal notation
o octal notation
x hexadecimal notation
u unsigned notation
c character notation
s string notation
f floating point notation

Each of these is used following a percent sign to indicate the type of output conversion, and between those two characters, the following fields may be added.

- left justification in its field
(n) a number specifying minimum field width
. to separate n from m
(m) significant fractional digits for a float
l to indicate a "long"

These are all used in the examples which are included in the program presently displayed on your monitor, with the exception of the string notation which will be covered later in this tutorial. Compile and run this program to see what effect the various fields have on the output. You now have the ability to display any of the data fields in the previous programs and it would be to your advantage to go back and see if you can display any of the fields anyway you desire.

LOGICAL COMPARES

Load and view the file named COMPARES.C for many examples of compare statements in C. We begin by defining and initializing nine variables to use in the following compare statements. This initialization is new to you and can be used to initialize variables while they are defined. The first group of compare statements represents the simplest kinds of compares since they simply compare two variables. Either variable could be replaced with a constant and still be a valid compare, but two variables is the general case. The first compare checks to see if "x" is equal to "y" and it uses the double equal sign for the comparison.

A single equal sign could be used here but it would have a different meaning as we will see shortly. The second comparison checks to see if "x" is greater than "z". The third introduces the "NOT" operator, the exclamation, which can be used to invert the result of any logical compare. The fourth checks for "b" less than or equal to "c", and the last checks for "r" not equal to "s". As we learned in the last chapter, if the result of the compare is true, the statement following the "if" clause will be executed and the results are given in the comments.

Note that "less than" and "greater than or equal to" are also available, but are not illustrated here. It would be well to mention the different format used for the "if" statement in this example program. A carriage return is not required as a statement separator and by putting the conditional clause on the same line as the "if", it adds to the readability of the overall program.

To be continued...

oooooooooo0000000000oooooooooo

Princeton University CoCo Listserver

For those of you who have access to E-Mail, you may care to subscribe to the CoCo listserver and automatic mailing list. Most of the discussions on the list are related to OS9/K, and is a good way to keep up to date with the latest developments from the USA. Be warned however, that there can be large volumes of mail on occasions. It may cause your mailer to convulse, or worse still, it may cause your SysAdmin to do the same!

GUIDE TO THE COCO MAILING LIST --- LAST UPDATED 3/05/90

by Paul E. Campbell <pecampbe@mtus5.BITNET>

1. PURPOSE OF THE LIST: COCO is an electronic mail discussion list that gives people on BITNET and ARPA a chance to discuss anything they wish related to any model of the Tandy Color Computer.

Some of the topics discussed on CoCo are: The Tandy Color Computer 3, and CoCo 1 and 2, OS-9 (Level I and Level II), 6809 programming, hardware, software reviews, source code exchange (public domain only), tips, hints, questions, rumors, and more!

* The list is run by the LISTSERV program, which also provides a unique DATABASE of CoCo programs and information available to members of the COCO list. These programs may be requested from the file server 24 hours a day, and will be sent to your account.

2. THE LIST: This is an open list, and is not edited. Anything you post will be sent out automatically to everyone subscribed to the list. The list is open to everyone, provided they behave properly. You are encouraged to post often to the list - an active mailing list is more interesting for everyone!

3. HOW TO ACCESS THE LIST: The list is run on a LISTSERV processor running on node PUCC on the BITNET network.

The following are different EMAIL addresses that may be used to get to PUCC. In each case, _____ represents the account on PUCC to which you are sending your message. The different accounts will be listed in a moment.

_____@pucc.BITNET
_____@pucc.Princeton.EDU
_____%pucc.bitnet@cunyvm.cuny.edu
rutgers!pucc.bitnet!_____
rutgers!pucc.princeton.edu!_____

The following accounts/userids should be put in the place of the blank for the following occasions:

COCO -- This is the list distribution account. Whatever you send to COCO will be posted automatically to everyone one else on the list. Use COCO when replying to a list message, posting information, or asking a question of everyone on the list.

LISTSERV -- This is the progam that runs the mailing list and file server. LISTSERV accepts commands only, which it will process. Some general LISTSERV commands are listed below to get you started using LISTSERV.

CHRIS -- This is the manager of the list server. Please only bother this guy if all else fails. (See below for contacting me, Paul Campbell, the list owner).

Choose the proper account, replace the _____ above with it, and MAIL to that address. If you're not sure which one to use, try them all, or your own versions of the above. For example, to MAIL a letter to everyone on the list

telling them what you think of the latest CoCo 3 software package you might type:

MAIL COCO@pucc.BITNET

And include your message text.

From now on, the USERIDs will simply be referred to as _____@PUCC. If you are not on BITNET, you will need to use one of the full addresses listed above, however.

Usually, you will be able to Reply to any mail messages you get from the list, and your responses will go back out to everyone on the list.

If you are still having trouble getting the list server to accept you, you can mail me, Paul Campbell, at the following addresses:

Bitnet: PECAMPBE at MTUS5
Internet: PECAMPBE@mtus5.cts.mtu.edu

4. ABOUT LISTSERV:

LISTSERV is a program written by Eric Thomas, and serves several major functions:

1) It processes mail sent to COCO and sends it to everyone on the list.
2) It keeps logs of those messages so they can be retrieved later.
3) Commands may be send directly to LISTSERV to do many functions without human intervention.
4) It is a file server, and will send out files and information upon request.
5) It creates a DATABASE from all past postings, which may be searched for any keyword or subject.

As mentioned above, COMMANDS only should be mailed to LISTSERV@PUCC. (BITNET users, note that you can use the interactive TELL LISTSERV AT PUCC msg command, replacing 'msg' with your command, for faster processing.)

When you send mail to LISTSERV, each of the lines in the mail body will be treated as a command. Some commands to get you started are:

HELP	- Sends you a Help Menu
REGISTER Your Real Name	- Registers your full name (not ID!)
SUBSCRIBE COCO Your Name	- Signs you up to list COCO
UNSUBSCRIBE COCO	- Removes you from the list
REVIEW COCO	- Reviews members & status

	of the list
INDEX COCO	- Lists all files available to list members
INDEX OS9	- Lists all available OS-9 only files
INFO ?	- Lists LISTSERV information available
INFO GEN	- General information about LISTSERV
INFO FILE	- Information about file server functions
INFO REFCARD	- Reference card of LISTSERV commands
INFO DATABASE	- Information about the DATABASE feature
SENDME filename filetype	- SENDs you the requested file.

You will receive confirmation from LISTSERV for any commands you send. Note that it is not necessary to send LISTSERV your userid or node. It will get that information from the mail header.

BITNET users should have no trouble sending MAIL to LISTSERV (just remember - one command per line, and no garbage - this is a program).

ARPA and UUCP users should be able to talk to ListServ as well, but if you can't (ie. you've waited for days and no response), and want a file or action to be taken, contact PECAMPBE.

5. MORE ABOUT FILE SERVERS...

If you do an INDEX COCO command, you will be sent a file of all current files on the COCO system. To get one of these files, send LISTSERV the command:

SENDME filename filetype COCO

If there is a file with a filetype of $PACKAGE, that means that you can obtain the whole set of files that deal with that subject with one command. For instance, there is currently a FORTH package for the CoCo available called DAC-Forth. To get all the files associated with it, tell LISTSERV:

SENDME DACFORTH PACKAGE COCO

Note that you do NOT include the "$" when requesting the

package.

NOTEBOOK files have a listing of all correspondence going through the list. The file COCO NOTEBOOK contains the first two weeks of the list. After that, files are kept in weekly logs, in the following format:

COCO LOGyymmW -- where yy is the Year, mm is the Month, and W is a letter for the week (A-E).

6. SUBMITTING FILES FOR THE SERVER:

Our database of programs is only as good as YOU make it. We need submissions. All submissions should be sent to PECAMPBE@MTUS5. All uploads and downloads which are program (not ASCII readable text), should use the COCO - USENET TRANSFER SYSTEM, called CUTS for short. You'll need the CUTS program to encode and decode these programs. Versions currently exist for CoCo BASIC, OS-9, and IBMPC. Send COCO and INDEX OS9 commands to see the current versions.

For more information on CUTS, send the command:

SENDME CUTS FORMAT

If on BITNET, please use the SENDFILE (or equivalent) command to send your text files or programs to PECAMPBE. Otherwise, use mail. You can put multiple CUTS files in one file. Please put a short explanation of what the program is on top of the CUTS code in your file.

7. SUMMING IT UP...

Enjoy the vast resource of the COCO lst, and please help make it grow! We need active contributors to both the list and the file server!

If you have any questions, please do not hesitate to contact me!

Also, tell your friends on the network about COCO, and send this file to them if you wish.

8. SPECIAL NOTE TO UUCP USERS:

The UUCP gateway here is unreliable. If you find you are not getting a response from the list, send me a message with a path, giving a nearby ARPA or BITNET node. (For instance: moss!yoursite!yourname@RUTGERS.EDU - in this case, Rutgers can be used for your articles, and they should get there fine.)

9. SPECIAL NOTE TO BITNET USERS:

You can use the interactive TELL LISTSERV AT PUCC to send your commands to LISTSERV, (or MSG LISTSERV AT PUCC, or

whatever variant exists on your system), and you also have the option of requesting what FORMAT you want the files in. Any time you send MAIL, you will get the requested file back as a MAIL file, unless you specify options. NETDATA, PUNCH, DISK DUMP, and other formats are available - for more information, give LISTSERV the command INFO FILE or INFO GENERAL or INFO REFCARD.

HAPPY HACKING! Enjoy the information and people who are on COCO!

oooooooooo0000000000ooooooooooo

Mac follies make us feel better

Mike Knudsen, writing on the Internet CoCo List, shows that we are not alone.

Today I attended a demo of some MIDI music hardware, using a Mac Plus (the biggest model that looks like a Mac). The grief two guys went thru getting the system up looked like a parody of every Coco Club meeting I've ever seen.

On top of the Mac sits an external fan to keep its cool. They feed several floppies to transfer to the (external) hard drive. About the 3rd disk the screen starts doing all sorts of random tweeds and plaids patterns -- just like a crashed Coco only not in living color.

They conlude that the MIDI application is too old a version for the latest Mac OS, so they reboot and feed more disks. The screen dialog boxes keep saying "swap this disk etc." and they're swapping disks in and out like a Vegas blackjack dealer. Crashes again, cardigans this time.

Third try they get it working, more swapping of copy-protected disks, and we hear some good music thru an external MIDI converter box. Yes, another add-on box to drag around.

I'll remember this day whenever I feel depressed about our "inferior" "toy" computer.

oooooooooo0000000000ooooooooooo

SCSI info for the MM/1?

Scott McGee asks*

On the MM/1, what exactly is the SCSI bus interface like? What connector(s) does it use? Is it SCSI 1 or 2? What exactly is the difference?

I assume that HD drivers are provided with the system for a SCSI hard drive. What drives/types of drives will work with it? Which won't? Are any drivers provided for other SCSI devices?

I assume that HD drivers are provided with the system for a SCSI hard drive. What drives/types of drives will work with it? Which won't? Are any drivers provided for other SCSI devices?

Any other info on SCSI apreciated.

Mark Griffith Replies*

It is a 50 pin dual-header connector on the I/O board. The bus is just the cable (ribbon cable internally) where you plug in various SCSI devices. The external connector has yet to be decided upon, but I think a DB-25 will be the choice. This will allow people to use relatively inexpensive Macintosh SCSI cables to connect to external devices.

The MM/1 SCSI is SCSI-1. SCSI-2 can be implemented later as the SCSI controller chip used is pin-for-pin compatible with the new SCSI-2 controller chip. The only thing that would need to be changed is the chip and the drivers.

The biggest difference between SCSI-1 and SCSI-2

is SCSI-2 allows for a bus width larger than the SCSI-1 8 bits. It also allows for some very very fast transfers across the bus.....up to 40 megaBYTES a second with a 32 bit bus. Of course, the MM/1 uses an 8-bit bus and will do so for some time. I don't see a change to a wider bus in the near future.

Any type of hard disk with an imbedded SCSI controller will work. Just about any SCSI drive you see sold today is one of these. All you need to do is plug it in after defining the SCSI device number, setup the software, and go. There is no upper size limit.

Oh....BTW....the MM/1 SCSI drivers now support partitioning drives....up to 16 partitions per drive if you want to. Easy to do also.

There are available (or very soon to be available) SCSI tape units and drivers for backing up. Carl Kreider has these tested and working fine. All we need now is to start selling them.

/\/\ark

oooooooooo0000000000oooooooooo

OSK PD programmes from Europe

We have been able to obtain a number of OSK public domain programmes from the EFFO group (European Forum For OS9). This is a group of OS9/K users who are located in Europe, and have their base in Zurich, Switzerland. For those who are interested, the Oz User Group can get copies of the files from the OS9 FTP site, with difficulty. These files will not, however be added to the User Group Library disks. The following information comes from the EFFO Group.

It is not practical to distribute all our forum programs on disk. While the forum is a chronological sample of pd files, the pd-disks will always be the newest releases. You may also find other programs on pd-disks which are too big to be distributed on a Forum Disk. In order to get a pd-disk you can pay the fee to our post-account, or send a cheque to one of our editors. Note carefully which disk you want on your order. For complete lists, contact Don Berrie.

No	Title	Revision	Date	Key words
0	Fortran	A	3/88	Real time fortran, huge documentation, and many examples.
1	Utilities I	A	5/90	system utilities (du, chown, phone, mail ...)
2	Communication I	A	5/90	ARC, KERMIT, ZOO, SHAR, UUEN-/UUDECODE ...
3	SB-Prolog	A	5/90	entire PROLOG system
4	XLisp 2.0	A	5/90	Xlisp - Good Documetation
5	Games I	A	5/90	tetris, nethack 2.2a, fortune cookies
6	TILE	A	5/90	fast FORTH system with quite big library (debugger, multitasking...), extensive documentation.
7	SC	A	5/90	spreadsheet calculator
8	EPHEM	A	5/90	astronomical program to calculate almost every interesting data about Planets and other objects in the sky (position, rise & set times, brightness...)
9	C++	A	3/91	GNU C, C++ Compilers and LIBG++ gcc_1.39.0, gpp_1.37.0, libgpp_1.37.0

Planned PD-disks: - Programmers Tools (YACC, LEX...)
- Little Smalltalk

If you are interested in obtaining any of these files, contact Don Berrie.

oooooooooo0000000000oooooooooo